FPGA SPIKING NEURAL PROCESSORS WITH SUPERVISED AND UNSUPERVISED SPIKE TIMING DEPENDENT PLASTICITY

A Thesis

by

SAI SOURABH YENAMACHINTALA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Peng Li
Co-Chair of Committee,	I-Hong Hou
Committee Member,	Rabi N. Mahapatra
Head of Department,	Dr.Miroslav M.Begovic

December 2018

Major Subject: Computer Engineering

ABSTRACT

Energy efficient architectures for brain inspired computing have been an active area of research with recent advances in the field of neuroscience. Spiking neural networks (SNN) are a class of artificial neural networks in which information is encoded in discrete spike events, closely resembling the biological brain. Liquid State Machine (LSM) is a computational model developed in theoretical neuroscience to describe information processing in recurrent neural circuits and can be used to model recurrent SNNs. LSM is composed of an input, reservoir and output layers. A major challenge in SNNs is training the network with discrete spiking events for which traditional loss functions and optimization techniques cannot be applied directly. Spike Timing Dependent Plasticity (STDP) is an unsupervised learning algorithm which updates synaptic weights based on time difference between spikes of pre synaptic and post synaptic neurons. STDP is a localized learning algorithm and induces self organizing behaviors resulting in sparse network structures making it a suitable choice for low cost hardware implementation. SNNs are hardware friendly as presence or absence of a spike can be encoded using a binary digit. In this research, SNN processor with energy efficient architecture is developed and is implemented on Xilinx Zynq ZC706 FPGA platform. Hardware friendly learning rules based on STDP are proposed and reservoir and readout layers are trained with these learning algorithms. In order to achieve energy efficiency, sparsification algorithm utilizing STDP rule is proposed and implemented. On chip training and inference are carried out and it is shown that with the proposed unsupervised STDP for reservoir training and supervised STDP for readout training, classification performance of 95% is achieved for TI corpus speech data set. Classification performance, hardware overhead and power consumption of the processor with different learning schemes are reported.

DEDICATION

To my grandparents, my parents and my professors.

ACKNOWLEDGMENTS

I would like to express my gratitude to my thesis advisor Dr. Peng Li for his continuous guidance through out the research. I would like to thank my thesis co-chair Dr. I-Hong Hou and committee member Dr. Rabi N. Mahapatra for being on my committee and providing constructive feedback for my work. I am also grateful to my friends Yu Liu and Yingyezhe Jin in the research group for their work in the area, and the help I have received from them. I express my appreciation for the faculty members of Texas A&M University for providing me excellent knowledge and experience in various areas related to my research. I would like to also thank my family and friends for encouraging me in this pursuit.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Dr. Peng Li [advisor] and Dr. I-Hong Hou of the Department of Electrical and Computer Engineering and Professor Dr. Rabi N. Mahapatra of the Department of Computer Science and Engineering. I have received continuous feedback from Dr. Peng Li which helped me improve my work throughout the research. Dr. I-Hong Hou and Dr. Rabi N. Mahapatra have provided valuable suggestions to analyze the certain aspects of the research. Neuron, synapse and unsupervised learning modules described in section 4.2 and 4.3 have been developed in collaboration with Yu Liu, PhD student in Dr. Peng Li's research group.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This work was supported by the National Science Foundation under Grant No. CCF- 1639995 and the Semiconductor Research Corporation (SRC) under Task 2692.001.

NOMENCLATURE

ANN	Artificial Neural Network
SNN	Spiking Neural Network
LSM	Liquid State Machine
STDP	Spike Timing Dependent Plasticity
FPGA	Field Programmable Gate Array
LIF	Leaky Integrate and Fire
LTP	Long Term Potentiation
LDP	Long Term Depression
CT	Classification Teacher
D-S^{2}TDP	Deterministic Supervised Spike Timing Dependent Plasticity
CaL-S^{2}TDP	Calcium modulated learning based on supervised spike timing dependent plasticity
CaS-S^{2}TDP	Calcium modulated sparsification based on supervised spike timing dependent plasticity
OE	Output Element
LE	Liquid Element
MSB	Most Significant Bit
AI	Artificial Intelligence
GPU	Graphic Processing Unit

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Background and Motivation

Computational power is one of the major requirements in a data centric era. Recent advances in science and technology have revolutionized the way computations are being performed. Zillions of bytes of data is being collected across the globe every second and various applications rely on this data to solve several complex problems. Processing high volumes of data requires high computational capability. Traditional Von-Neumann architectures are turning out to be inefficient and usage of large number of devices is resulting in high costs. Semiconductor industry is moving towards the end of Moore's law. Increasing computational power through an increase in the number of processors on a chip is no longer a feasible solution. Transistor sizes are being pushed to their fundamental physical limits and increase in the number of transistors on a chip results in an increased power consumption. These factors motivated researchers to search for alternate ways to efficiently handle large volumes of data. One such motivation is obtained from biological brain and its computational efficiency. Brain is one of the most complex organs known and is highly efficient in processing large volumes of data at a very high speed, consuming low energy. These properties of biological brain inspired development of computational units capable of processing data similar to biological nervous system.

Processing huge volumes of data requires dedicated hardware platforms functioning at high speed with high degree of accuracy. Hardware architects have been developing high speed computing systems with a goal to keep Moore's law alive by overcoming power walls to a great extent. However, a general purpose processor is not very efficient to carry out specific set of computations. Recent era has witnessed an increase in the use of graphic processing units (GPUs) to deploy several neural network tasks as these dedicated highly parallel architectures can process large amounts of data at a very fast rate. Researchers employ a large number of GPUs to train their neural network models to achieve a high degree of efficiency. As the number of computing systems increase,

power consumption will increase which will in turn result in an increasing need for cooling systems. Even with such high computing power, a GPU is still inferior compared to biological brain in terms of both size and speed. Power consumption by a biological brain does not require a cooling system inside living organisms. An insect whose brain is similar to the size of a pea is able to perform tasks with greater degree of accuracy compared to the current computing platforms. This motivates the need to develop specialized architectures which can mimic computational efficiency of biological brain. Two brain inspired computing paradigms have emerged in the field of artificial intelligence(AI), namely artificial neural networks (ANN) and spiking neural networks.

The idea of brain inspired computing dates back to over fifty years. The first generation of ANNs consisted of a simple computational model proposed by MuCulloch-Pitts. According to this model a neuron, fundamental information processing unit of a neural network, sends out an output signal if the sum of its input signals exceeds a threshold. The output from the neuron is binary. Although this model is very simple, it has been used to construct some powerful neural networks like multi layer perceptrons capable of performing complex tasks. In the second generation of ANNs, the threshold function is replaced by a continuous activation function allowing continuous input and output. Some of the most commonly used activation functions are sigmoidal function, rectified linear unit etc. Feed forward and recurrent neural network architectures have been developed using these activation functions. The first two generations of neural networks employ rate encoding scheme. In such an encoding, if a neuron fires N spikes in a time interval T then the output of neuron is proportional to $\frac{N}{T}$. Several architectures such as convolutional neural networks, recurrent neural networks, support vector machines etc have been proposed to solve increasingly complex tasks with high degree of accuracy. Inspite of high degree of complexity, these architectures are functionally very distinct from biological brain and consume high amount of power.

Advances in the field of computational neuroscience and neurobiology indicated spatio-temporal information encoding in biological systems in contrast to rate encoding. This led to the development of spiking neural networks, a class of bio-inspired computational models where neurons

communicate with each other through a sequence of spikes. This third generation of neural networks consider spatio temporal information to process the input signals. Research in neuroscience has demonstrated that humans respond to any changes in input at a very fast rate. It takes less than 100ms to recognize a change and take an appropriate action. For example, to detect a visual change a signal has to travel from retina through optic nerve to reach temporal lobe during which a signal goes through atleast ten stages of processing leaving about 10ms for each stage. This time window is too small to employ an averaging mechanism like rate coding which motivates the presence of temporal encoding in biological brain [3].

Fundamentally, ANNs and SNNs differ in the way information is encoded. However, SNNs have high biological plausibility compared to ANNs owing to the way in which information is processed. This biological plausibility can achieve energy efficiency while providing high computational ability. In terms of performance, ANNs are superior compared to SNNs owing to the complex encoding and learning mechanisms of SNNs along with lack of efficient computing architectures. Computing in SNNs is achieved using a reservor computing model. Liquid State Machine is a class of reservoir computing which operates on spiking neurons. The architecture of LSM is highly efficient for hardware implementation and provides a good trade off with computing power of SNN. This research utilizes computing capability of spiking neurons combined with hardware efficiency of LSM to develop energy efficient neuromorphic architectures.

1.2 Neurons in Silicon

Human brain is the most complex and a fascinating organ to study. With all the advancements in the field of neurosience and neurobiology, very little is known about the way in which brain processes information. Several attempts have been made to mimic architecture of biological brain on silicon. Caver Mead at California Institute of Technology developed the first silicon architecture capable of processing visual information and coined the term neuromorphic systems [4]. Ever since tremendous efforts have been made to develop efficient hardware systems capable of replicating biological architectures. Developing energy efficient neuromorphic processors has been an active area of research for past few years. With spatio temporal information processing, SNNs

became a promising class of ANNs capable of providing energy efficient solution to the field of neuromorphic systems. Application of SNNs to solve various real world applications is currently limited by complex learning mechanisms and lack of efficient processor architectures. It is difficult to develop efficient techniques capable of processing information in time domain compared to techniques which can process information in frequency domain. Spatio-temporal information processing of SNNs makes it a suitable architectural choice for applications such as speech processing. Data sets available today have been captured by digital systems which work with a completely different mechanism compared to biological systems. Processing in biological systems happens in a continuous domain while almost all the data sets available today have been captured and are being processed in digital domain. It is a great challenge to mimic a biological system processing data in continuous domain to a digital system. Lack of standard data sets is another limiting factor for the ability of SNNs to solve several real world tasks. IBM TrueNorth, Stanford Neurogrid, Intel Lohihi are some of the recent VLSI implementations of spiking neural networks. Several SNN architectures are targeted towards FPGA implementation. Perceptron readout layer with delta prop trained reservoir layer based VLSI architecture is also implemented [5]. Most of these processors support off-line training and on chip inference, owing to the complexity associated with training a neural network. As SNNs are biologically plausible it is feasible to develop low power on chip training processors with an increased speed of learning resulting from parallel processing of spikes by all neurons in a layer. [6] introduces unsupervised STDP training which is used to tune reservoir synapses while read out synapses are trained using change in calcium concentration of synapses. This approach restricts STDP only to reservoir layer and also requires full connectivity between reservoir and readout layer synapses. [7] introduces sparsification of synapses in readout layer but does not maintain a good performance-sparsification trade off and no STDP in readout layer. [6] introduces a calcium modulated supervised STDP in readout layer and STDP based sparsification. This method offers a good trade off between hardware cost and classification performance making it a good choice for hardware implementation.

1.3 Challenges in SNNs

Several challenges need to be addressed to develop an efficient hardware architecture for a spiking neural network. Learning mechanisms involved are complex and are in continuous domain. It is essential to develop hardware friendly learning mechanisms which are feasible for digital architectures. Achieving high computational power with low energy consumption is one of the major challenges which needs to be addressed. Existing architectures implementing SNNs do not support learning on the chip. On-chip learning improves efficiency and speed of learning by utilizing parallel computing structures. These challenges are addressed in this research.

Primary goal of this research is to develop an energy efficient neural network processor employing principles of biological information processing. To achieve this goal, hardware friendly STDP learning algorithm is used to train neural network architecture. A major component of a nervous system responsible for transmitting information from one neuron to another is a synapse. There exists several billions of synapses in biological brain interconnecting neurons in complex ways. As these synapses are abundant in existence, energy efficient hardware architectures can be developed by disabling those synapses which are irrelevant to the tasks under consideration. To achieve high performance and energy efficiency, a supervised learning algorithm along with a sparsification algorithm is proposed. This supervised learning algorithm, termed as calcium modulated learning based on supervised STDP (CaL-S^{2}TDP) employs a supervisory signal to help the neural network learn the features in the input patterns accurately and thus make a correct inference. Synapse weights are limited to have small bit resolutions to achieve small hardware overhead and low energy due to which possibility of weight saturation is high. To avoid this problem, a stochastic weight update scheme is proposed. In order to disable synapses which do not affect learning performance and thus achieve energy savings, calcium modulated sparsification algorithm based on supervised STDP (CaS-S^{2}TDP) is proposed. A unified training mechanism is described to train neural network using both CaL-S^{2}TDP and CaS-S^{2}TDP.

2. SPIKING NEURAL NETWORKS

Spiking neural networks and their computational models are based on several complex mechanisms involved in the functioning of biological brain. Brain is composed of millions of information processing units called neurons, forming an intricate network. It is essential to understand the structure and function of neurons to develop computational models for modeling spiking neural networks. This section begins with describing the structure and function of a spiking neuron and its computational models. An SNN is developed by a random interconnection of these neurons and liquid state machine, a computational model for SNNs is also discussed in this section.

2.1 Spiking Neurons

A neuron is a fundamental information processing unit in biological brain. Figure 2.1 shows the structure of a neuron. Physiologically, a neuron can be divided into three distinct regions namely, dendrites, soma and axon. A neuron receives input signals from its neighbors through dendrites. These signals are processed in the cell body, soma which is a non-linear processing system. The incoming signals from dendrites change the membrane potential of neuron and if the membrane potential exceeds certain threshold then neuron generates a spike, known as action potential. Action potential traverses down the axon which branches out in several directions. There are some neurons whose axonal lengths are in the order of hundreds of meters. These branches come in contact with adjacent neurons through a junction called synapse. It is through synapse, electrical signals are transmitted from one neuron to another. Thus, a neuron can be visualized as a system consisting of an input (dendrites), processing unit (soma) and an output (axon). A neuron which transmits action potential is known as presynaptic neuron while a neuron which receives action potential is known as postsynaptic neuron. A neuron at rest is associated with a membrane potential known as resting potential, V_{rest}. Figures 2.2 and 2.3 show a simplified view of a neuron and interaction between presynaptic and postsynaptic neurons through synapse respectively.

Generation and propagation of an action potential depends on the exchange of various ions

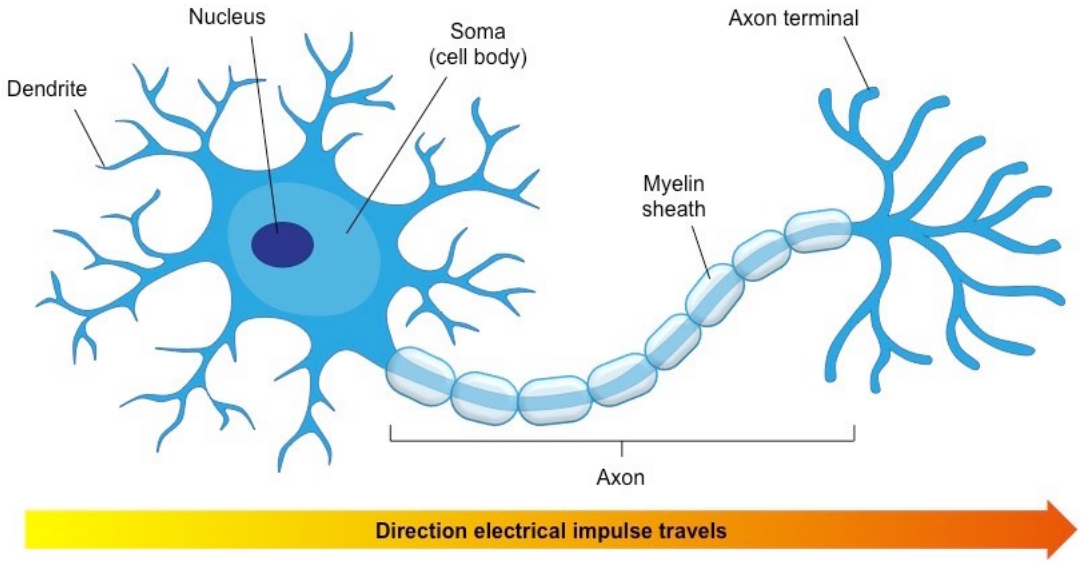

Figure 2.1: Structure of a neuron. Reprinted from [1]

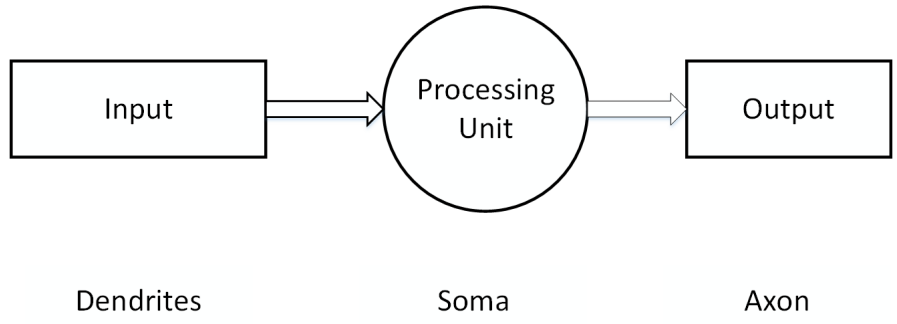

Figure 2.2: Simplified view of a neuron

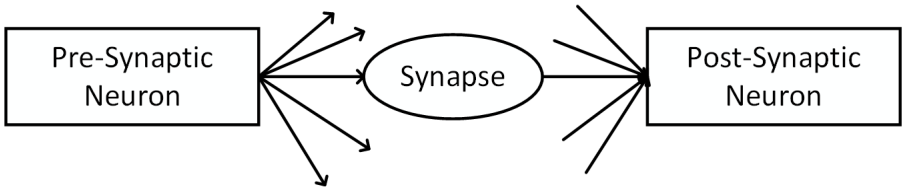

Figure 2.3: Interaction of presynaptic and postsynaptic neurons

such as sodium, potassium, calcium, chlorine etc across neuronal membrane. An ion of interest, responsible for transfer of action potential across neuronal synapse is calcium. Voltage sensitive

calcium ion channels present on presynaptic neuronal membrane open due to change in membrane potential. Concentration gradient of calcium ions across plasma membrane results in calcium ions rushing into pre-synaptic terminal. These calcium ions then bind to proteins such as synaptotagmin which promote fusion of synaptic vesicles (organelles responsible for release of chemical substances known as neurotransmitters) and thus, release of neurotransmitters into synaptic cleft. If intra-cellular calcium concentration is high, it results in over excitation of neural circuits as synaptic vesicles continuously fuse with plasma membrane releasing neurotransmitters. Thus, the dynamics of calcium ion concentration plays an important role in the generation of spike by a neuron. This property of calcium ion concentration is used in this research to control the spiking rate of a neuron.

A neuron can be modeled as either excitatory or inhibitory depending on whether it excites or inhibits a postsynaptic neuron from firing an action potential. Biologically speaking, this behavior is attributed to different types of neurotransmitters. Certain neurotransmitters like dopamine allow ion channels on neuronal membrane to open, aiding in generation of action potential while neurotransmitters like gamma amino butyric acid close ion channels on neuronal membrane inhibiting postsynaptic neuron activity. This property of neurotransmitters is taken into account in this research and is reflected in the neuron and synapse models discussed in the sections below.

2.2 Action Potentials or Spikes

Action potential or spike generated by a neuron is a voltage signal which is typically 100mv in amplitude and 1-2ms in duration. Ion channels spread across axonal membrane act as repeaters ensuring signal fidelity along the length of axon. As the shape of all spikes generated by neurons is identical, information is encoded in the timing and number of spikes generated. A neuron will not be able to generate spikes continuously. Minimum amount of time required for a neuron between generation of consecutive spikes is known as absolute refractory period. Hence, spikes are discrete events and a sequence of spikes generated by a neuron constitutes a spike train.

Shape of an action potential or spike can be described using changes in the properties of neuronal membrane. Figure 2.4 depicts the shape of an action potential. A neuronal membrane un-

dergoes several changes when an action potential is generated. These changes can be divided into several stages such as:

1. Rising Phase - Nerve membrane is depolarized to an extent that membrane potential starts to become positive with respect to external medium.

2. Overshoot Phase - During this phase, neuronal membrane potential is positive with respect to outer membrane. After reaching a peak value, action potential eventually enters into falling phase due to change in permeability of membrane to specific ions.

3. Falling Phase - During this phase, membrane potential is repolarized to a value lower than resting potential.

4. Undershoot Phase - Membrane potential slowly returns to resting potential during this phase.

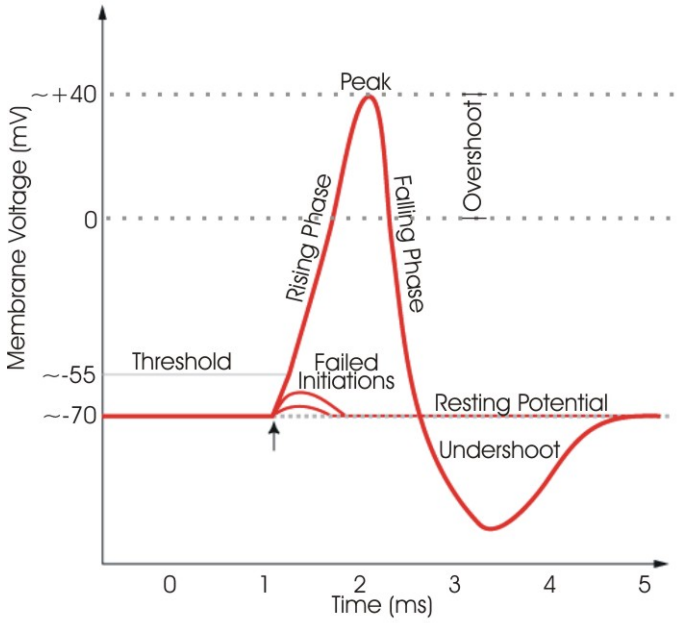

Figure 2.4: Shape of action potential. Reprinted from [2]

2.3 Computational Models

Computational models at different levels of abstraction are available to model a spiking neuron. As discussed in section 2.1, an action potential is a result of currents passing through ion channels resulting in changes in membrane potentials. Hodgkin and Huxley [8] performed a series of experiments on giant squid axon and developed a model describing the dynamics of ion channel currents in terms of differential equations. These models can replicate the behavior of neuron with high degree of accuracy but are too complex to analyze and implement in hardware. As a result, several simple models have been proposed which abstracts the dynamics of ion channels using resistive and capacitive elements. The neurons in this research are modeled using leaky integrate and fire (LIF) model.

2.3.1 Leaky Integrate Fire Model

A spiking neuron in LIF model is described as an RC circuit consisting of a capacitor in parallel to a resistor as shown in figure 2.5. Using this model, dynamics of membrane potential can be

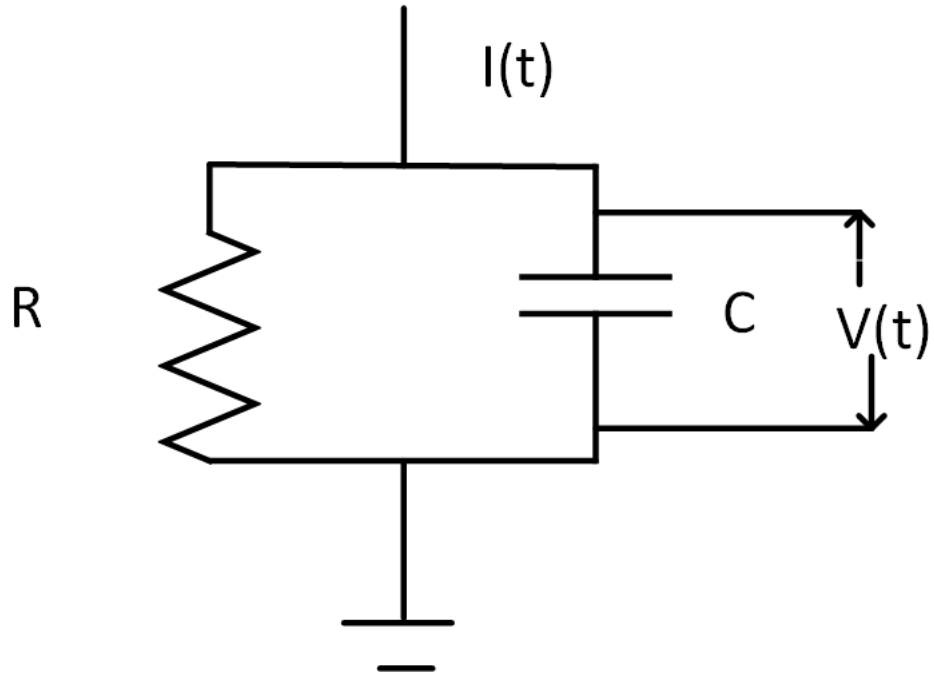

Figure 2.5: Neuron in LIF model

described by the following equation

$$\frac{dv_m}{dt} = -\frac{v_m}{\tau_m} + \frac{I(t)}{C} \tag{2.1}$$

The above equation models an isolated neuron stimulated by an external current I(t). However, in a network of neurons, each neuron is connected to several neighboring neurons with synapses. Current $I(t)$ will be a summation of all pulses received from each of the presynaptic neurons. Let t_{ij} represent j^{th} time instant during which a spike is received from a presynaptic neuron through a synapse of synaptic weight w_{mi} and let d_i denote delay associated with synaptic transmission and S(.) be synaptic transformation function. Current pulse received from presynaptic neuron i is described by the following equation

$$I_i(t) = \sum_j w_{mi} s(t - t_{ij} - d_i) \tag{2.2}$$

Integrating the current from all the pre-synaptic neurons, dynamics of membrane potential variation can be described by the following differential equation

$$\frac{dv_m}{dt} = -\frac{v_m}{\tau_m} + \sum_i \sum_j w_{mi} s(t - t_{ij} - d_i) \tag{2.3}$$

Supervised learning is a form of learning mechanism in which information about the output is provided to the learning element in the form of a teacher signal. In the case of spiking neurons, this teacher signal is a current induced into the neuron from an external source. Let $i_t(c)$ denote the current induced by teacher signal into the neuron. This current is expressed as a function of calcium concentration c. Including this term in the equation (2.3) gives

$$\frac{dv_m}{dt} = -\frac{v_m}{\tau_m} + \sum_i \sum_j w_{mi} s(t - t_{ij} - d_i) + i_t(c) \tag{2.4}$$

Equation (2.4) can be expressed in the form of a difference equation to make it compatible for hardware implementation. The resulting equation is expressed as follows

$$V_m^n = V_m^{n-1} - \frac{V_m^{n-1}}{\tau_m} + \sum_i \sum_j W_{mi} S(T^n, T_{i,j} + D_i) + I_t^{n-1} \tag{2.5}$$

The subscripts indicate discrete time steps.

A neuron fires when its membrane potential exceeds threshold voltage V_t, after which its membrane potential is reset to V_{rest}. Neuron goes through an absolute refractory period τ_{refrac} after each spike is fired during which it cannot fire a new spike. The dynamics of calcium concentration c is modeled by the following equation

$$\frac{dc}{dt} = -\frac{c}{\tau_c} + \sum_i \delta(t - t_i) \tag{2.6}$$

where τ_c is the time constant for first-order dynamics of calcium concentration c and i is the index of spikes emitted from the neuron itself.

2.3.2 Synapse Models

A synapse is a computational unit responsible for transferring a spike from presynaptic to postsynaptic neuron. Each synapse has an associated weight which influences the decision of a neural network. Combined weight of all the synapses in a neural network is an abstraction for memory of the network. Choice of synapse model has a great influence on memory of the network and thus, overall performance. It has been shown in [9] that a second order synapse model yields good performance compared to a first order and an impulse synaptic response. Moreover, [9] shows that second order model is hardware friendly in terms of its implementation. Equation 2.7 describes the dynamics of neuronal membrane voltage using dynamic second order synaptic model.

$$\frac{dv_m}{dt} = -\frac{v_m}{\tau_m} + \sum_i \sum_j \frac{w_{mi} e^{\frac{t-t_{ij}-d_{ij}}{\tau_1^s}} H(t - t_{ij} - d_{ij})}{\tau_1^s - \tau_2^s} + - \sum_i \sum_j \frac{w_{mi} e^{\frac{t-t_{ij}-d_{ij}}{\tau_2^s}} H(t - t_{ij} - d_{ij})}{\tau_1^s - \tau_2^s} \tag{2.7}$$

where τ_1^s and τ_2^s are time constants of second order response. The values of time constants are chosen to be a power of 2, for the ease of hardware implementation.

Equation 2.7 can be optimized as described in [9] and is reduced to a simpler form for hardware implementation.

$$V_m(t) = V_m(t-1) - \frac{V_m(t-1)}{\tau} + \frac{EP - EN}{\tau_{EP} - \tau_{EN}} - \frac{IP - IN}{\tau_{IP} - \tau_{IN}} \tag{2.8}$$

where EP, IP, EN and IN are state variables of second order responses and τ_{EP}, τ_{IP}, τ_{EN},τ_{IN} are their corresponding time constants respectively. A second order variable X is modeled by the following equation

$$X(t) = X(t-1) - \frac{X(t-1)}{\tau_X} + \sum w_i Y(i) \tag{2.9}$$

for $X = \{EP, EN, IP, IN\}$

In equation 2.9, w_i is the weight of the i^{th} synapse and $Y(i)$ is 1 if presynaptic neuron fires a spike at time t and 0 if there is no spike.

2.4 Liquid State Machine

Liquid state machine is a computational model proposed in [10] to model spiking neural networks. An LSM consists of an input layer, reservoir layer and an output or readout layer. The number of neurons in the reservoir are much higher compared to the neurons in input layer. As a result, input spike train is projected into a high dimensional space by the reservoir layer. Reservoir layer is composed of a recurrent network of neurons randomly connected to each other. A spike from the input layer creates a disturbance in the reservoir layer which is propagated from the point of disturbance towards readout layer, just as a ripple propagates in a pond from the point of disturbance. Readout layer is fully connected to the reservoir layer and the number of neurons in readout layer is equal to the number of input classes. Figure 2.6 depicts a model of LSM.

Neurons in the reservoir are arranged in the form of a grid of size $l * b * h$ and each neuron is randomly connected with other neurons through synapses such that neurons which are closer

together have a higher probability to be connected. The probability of a synapse between two reservoir neurons N_a and N_b is given by

$$P_{synapse}(N_a, N_b) = kexp(\frac{-D^2(N_a, N_b)}{m^2}) \tag{2.10}$$

where k and m are appropriately chosen constants.

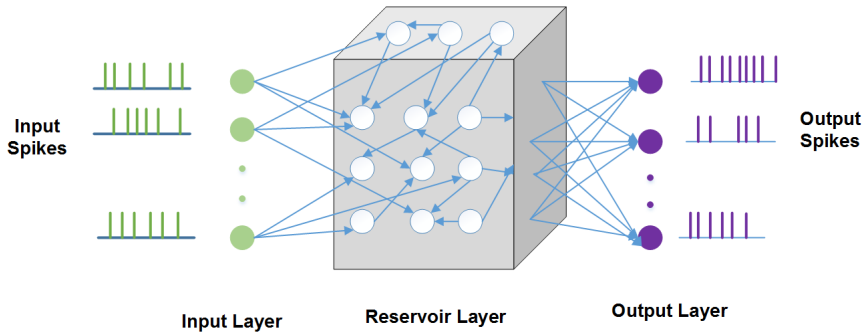

Figure 2.6: A model of liquid state machine

3. LEARNING IN SPIKING NEURAL NETWORKS

Learning is a process of adjusting the synaptic weights in order to optimize the performance of neural network for a given task. A procedure used to achieve such an optimization process is known as learning rule. There exists different kinds of learning in the theory of AI such as supervised, unsupervised, reinforcement learning etc. Simplest form of learning in SNNs are a result of large set of experiments carried out by Hebb on synaptic plasticity. According to Hebb, when a neuron A continuously excites a neuron B, then the synaptic strength between these two neurons increases such that neuron A develops to be a potential neuron responsible for firing neuron B. This principle is known as Hebbian learning. Long term potentiation (LTP), a persistent increase in synaptic strength and long term depression (LDP), a persistent decrease in synaptic strength are enhancements over basic Hebbian learning rule. This section describes spike timing dependent plasticity learning mechanism along with several enhancements and hardware optimizations to this learning rule.

3.1 Spike Timing Dependent Plasticity

Spike Timing Dependent Plasticity is an unsupervised Hebbian based learning rule which controls the plasticity of synapses based on temporal difference in spiking events of pre-synaptic and post-synaptic neurons [11]. A post-synaptic neuron, j is connected to several pre-synaptic neurons. For a given pre-synaptic neuron, i synaptic weight update Δw_{ji} is a function of temporal difference $\Delta t_{ji} = t_j - t_i$ between the spike pair. If a pre-synaptic neuron fires before post-synaptic neuron, synaptic weight increases (LTP). If a post-synaptic neuron fires before a pre-synaptic neuron, synaptic weight decreases (LDP).Thus, the strength of synaptic weight is a function of correlation between firing activities of presynaptic and postsynaptic neurons. The weight update in STDP learning can be expressed mathematically as

$$\Delta w_{ji}^{+} = A_{+}(w).exp(-\frac{|\Delta t_{ji}|}{\tau_{+}}) \tag{3.1}$$

$$\Delta w_{ji}^{-} = A_{-}(w).exp(-\frac{|\Delta t_{ji}|}{\tau_{-}}) \tag{3.2}$$

where Δw_{ji}^{+} and Δw_{ji}^{-} indicate change of synaptic weights due to LTP and LDP respectively, τ_{+},τ_{-} are time constants, A_{+} and A_{-} determine strength of LTP and LDP respectively.

STDP has an inherent self-organizing behavior capable of inducing sparsity in the network topology through introduction of competition among synapses [12]. This sparse nature of the network is utilized to construct energy efficient processor architectures described in section 4. Figure 3.1 plots STDP characteristics. To maintain stable network dynamics, only excitatory synapses are tuned.

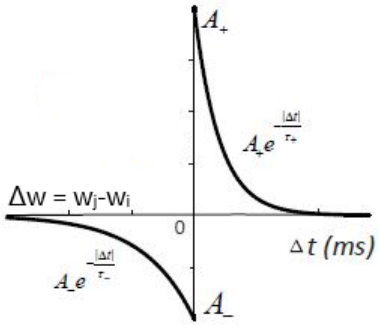

Figure 3.1: STDP characteristics

3.2 Calcium modulated learning in readout neurons

A readout neuron is expected to fire maximum number of spikes if it is a representative of input class while the remaining readout neurons are expected to have as low activity as possible. If an undesired neuron generates a spike, its activity can be reduced by reduction of synaptic weight. On the other hand, if a desired neuron does not generate a spike its activity can be increased by synaptic potentiation. Let u_p and u_d represent the present and desired activity of a readout neuron and u_T be the threshold firing activity which determines the firing rate of a readout neuron. If the readout neuron is a desired neuron and its current activity is less than the threshold, synaptic

potentiation must occur. If the neuron is not a representative of input class and its firing activity is above threshold, synaptic depression must occur. Expressing this mathematically

$$w_i \rightarrow w_i + \Delta w \textit{ if } u_p < u_T + \Delta u \textit{ and } u_d > u_T \tag{3.3}$$

$$w_i \rightarrow w_i - \Delta w \textit{ if } u_p > u_T + \Delta u \textit{ and } u_d < u_T \tag{3.4}$$

where Δu is a threshold which allows the learning process to be driven by correctly classified data. For hardware implementation, discrete synaptic weights of finite resolution are used. To avoid saturation of weights during learning process, effective learning rate is reduce by introduction of stochastic weight update scheme.

$$w_i \rightarrow w_i + \Delta w \textit{ with prob } p_+ \textit{ if } u_p < u_T + \Delta u \textit{ and } u_d > u_T \tag{3.5}$$

$$w_i \rightarrow w_i - \Delta w \textit{ with prob } p_- \textit{ if } u_p > u_T + \Delta u \textit{ and } u_d < u_T \tag{3.6}$$

As discussed in section 2, calcium ion concentration of a neuron is a good indicator of its firing activity. Replacing firing activity u with calcium concentration c, learning rule can be restated as

$$w_i \rightarrow w_i + \Delta w \textit{ with prob } p_+ \textit{ if } c_p < c_T + \Delta c \textit{ and } c_d > c_T \tag{3.7}$$

$$w_i \rightarrow w_i - \Delta w \textit{ with prob } p_- \textit{ if } c_p > c_T + \Delta c \textit{ and } c_d < c_T \tag{3.8}$$

Learning in a practical neural network depends only on the present firing rate and is independent of desired firing rate. To further inhibit synaptic saturation, two stop learning regions are proposed based on calcium concentration. These regions are $c_p > c_T + \Delta c$ and $c_p < c_T - \Delta c$. Learning occurs only when calcium concentration of a neuron is in the region $c_T - \Delta c < c < c_T + \Delta c$. Synaptic potentiation occurs when $c_T < c < c_T + \Delta c$ and depression occurs when $c_T - \Delta c < c < c_T$. Figure 3.2 shows various learning regions based on calcium ion concentration. Improved learning rule can be expressed mathematically as follows

$$w_i \rightarrow w_i + \Delta w \textit{ with prob } p_+ \textit{ if } c_T < c_p < c_T + \Delta c \tag{3.9}$$

$$w_i \rightarrow w_i - \Delta w \textit{ with prob } p_- \textit{ if } c_T < \Delta c < c_p < c_T \tag{3.10}$$

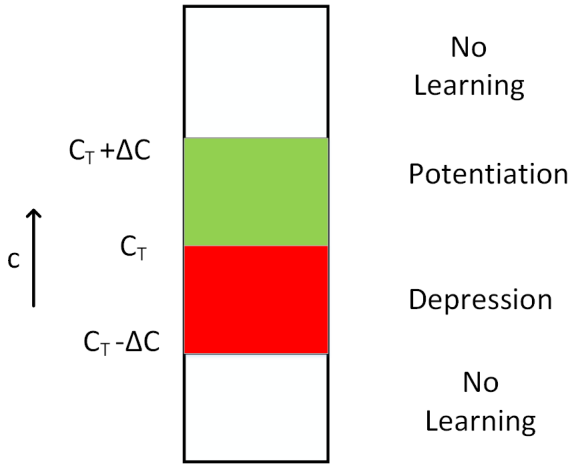

Figure 3.2: Learning regions based on calcium concentration

A teacher signal is provided to the readout neurons which will further increase the firing activity of desired neurons and inhibit the activity of undesired neurons. This teacher signal modulates the activity of readout neurons such that for a desired neuron calcium concentration is driven to $[c_T, c_T + \Delta c]$ and for an undesired neuron calcium concentration is driven to $[c_T - \Delta c, c_T]$.

3.3 Unsupervised STDP for reservoir training

Energy efficiency can be achieved by utilizing self-organizing behavior of STDP algorithm in training of reservoir neurons. However, continuous nature of STDP characteristics and corresponding weight updates are not suitable for realization in digital hardware. A major challenge involved in discritization of these parameters is the choice of bit resolution. Having a higher resolution closely approximates continuous domain, but is not energy efficient. On the other hand, low bit resolution can hurt the learning performance.In this research, a data-centric approach is adopted to

discritize STDP characteristics and weight updates. The data-centric discritization approach aims

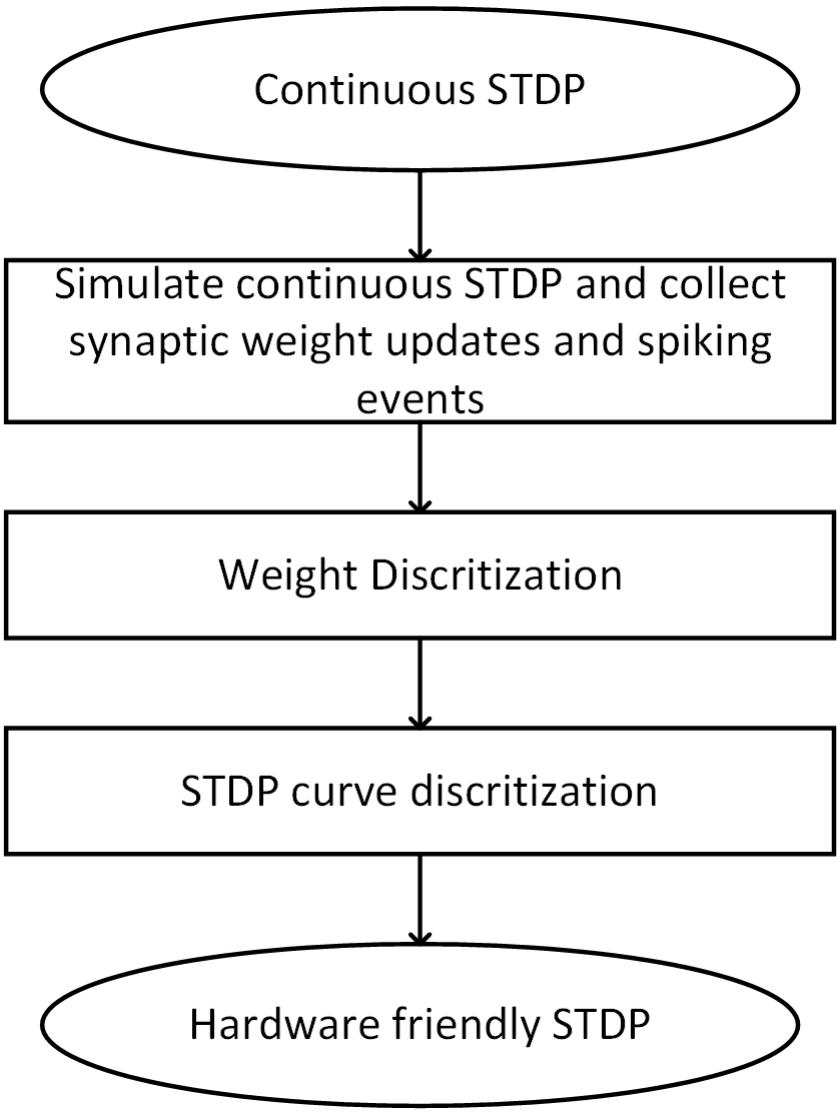

Figure 3.3: Proposed data centric approach to hardware friendly STDP

to discritize continuous weights such that their equilibrium distribution is maintained. Given temporal difference Δt_{ji} and continuous weight change Δw_{ji}^{c}, discritized STDP characteristics have to match the corresponding synaptic weight update in the continuous domain. The procedure for discritization can be divided into four stages as shown in Figure 3.3.

Simulation of Continuous STDP: Reservoir is simulated with given set of inputs and observed

weight changes and spike events are recorded as a set of four values: $(\Delta t_k, \Delta w_k^c, w_{prev,k}^c, w_{next,k}^c)$. $k \in [1, N]$. N such observations are recorded.

Weight Discritization: Given B bits to represent discritized weights, it is necessary to minimize the representation error of each continuous weight w_i^c. For a given continuous weight w_i^c and a set of discrete weights $D = \{w_l^d\}$, $l \in [1, 2^B]$, the goal is to choose a value from the set D which minimizes squared error between w_i^c and a given w_l^d. This minimization is carried out over all values of w_l^d. This can be mathematically expressed as follows

$$\begin{aligned} \min_{w_l^d} \quad & \textstyle\sum_l \min_{w_l^d} (w_k^c - w_l^d)^2 \\ \text{subject to} \quad & w_l^d \in [w_{min}, w_{max}], \forall l \in [1, 2^B] \end{aligned} \tag{3.11}$$

As the search space is small for smaller values of B, the above optimization problem can be solved easily.

Discritizing STDP: After obtaining optimal discrete weights, STDP curve has to be discritized to obtain discrete levels of time and weight updates. This is achieved by mapping $\{\Delta t_k, w_{prev,k}^d\}$ directly to a new weight w_{next}^d using a look up table (LUT) approach. An appropriate value of time step is chosen for the network and LUT is indexed by Δt and w_{prev}^d to obtain new weight, where Δt is a multiple of chosen time step. Each entry in the LUT serves as a discritized weight obtained under proposed STDP rule. Optimal LUT entries are obtained as follows. Using the quantized weights obtained from weight discritization step, $\{\Delta t, w_{prev}^c, w_{next}^c\}$ are mapped to $\{\Delta t, w_{prev}^d, w_{next}^c\}$ by choosing w_{prev}^d close to w_{next}^c. Let L_{mn} be the corresponding entry of w_{next}^c in the LUT which is indexed by $(\Delta t, w_{prev}^d)$. The goal is to discritize the LUT entries so as to minimize the aggregated error over all w_{next}^c in L_{mn}. This optimization problem can be mathematically expressed as :

$$\begin{aligned} \min_{L_{mn}} \quad & \textstyle\sum_k (w_{next,k}^c - L - ij)^2 \\ \text{subject to} \quad & L_{mn} \in [w_1^d, w_{2^B}^d] \end{aligned} \tag{3.12}$$

3.4 Supervised STDP for readout training

A supervised learning algorithm provides information about the corresponding class label, through a teacher signal aiding in the classification process. In case of spiking neural networks, a neuron with highest firing frequency indicates the corresponding class label of input. As such, the job of teacher signal is to maximize the firing frequency of desired neuron while inhibiting the activity of undesired neurons. This problem can be expressed mathematically as follows :

$$\begin{aligned} \max_{f_j^i} \quad & \textstyle\sum_{i=1}^{n}(f_{c(i)}^i(X_i, W) - \sum_{j\neq c(i)}^{C} f_j^i(X_i, W)) \\ \text{subject to} \quad & f_j^i \geq 0 \end{aligned} \tag{3.13}$$

where N is the total number of input samples, C is the total number of input classes, X_i is the i_{th} sample with $c(i)$ being its class label. f_j^i is the firing frequency of j^{th} readout neuron under i^{th} input and W is the weight vector of readout synapses.

Equation 3.5 tries to maximize the distance of firing rate between desired and undesired neurons so as to minimize the classification error over the entire training set. The above optimization problem appears to be complex to solve mathematically. It can be avoided by exploiting local weight update characteristics of STDP algorithm. Based on this approach, deterministic supervised STDP algorithm (D-S^{2}TDP) is proposed.

The motivation for D-S^{2}TDP algorithm is from the learning mechanism of STDP rule which tries to strengthen or weaken synaptic strength between two neurons based on their relative firing time. This can be used to control the firing activity of desired neurons through an introduction of a classification teacher (CT) signal which serves as supervisor. The function of this signal is to induce enough current into the desired neuron so that it can fire more frequently. This increases the number of causal firing events for the desired neuron. As such, STDP strengtens the synaptic weight. Due to higher synaptic weight, firing activity of desired neuron will further increase. CT signal also helps in making learning process robust by increasing the synaptic weight of desired neuron quickly, thereby reducing the classification error.

Figure 3.4 and 3.5 show the learning process of desired and undesired neurons in $D-S^2TDP$.

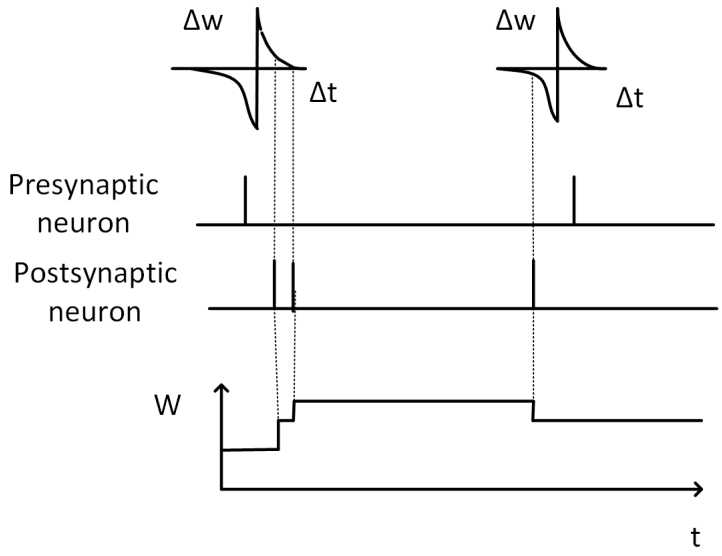

Figure 3.4: Training of desired neurons using D-S^{2}TDP

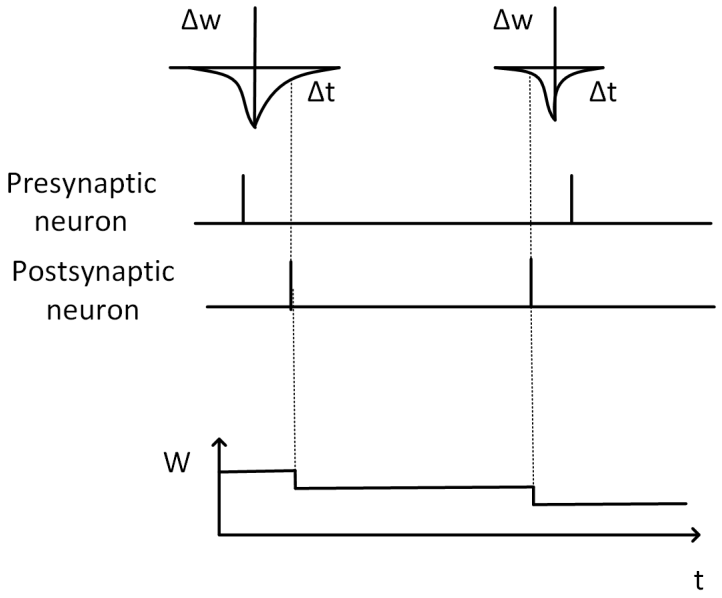

Figure 3.5: Training of undesired neurons using D-S^{2}TDP

Positive correlation between presynaptic and postsynaptic spiking events results in increase of synaptic weight and introduction of current through teacher signal invokes an additional spike which further increases synapse strength. A negative correlation will reduce synaptic strength.

In case of undesired neurons, a positive or negative correlation results in reduction of synaptic strength with teacher signal trying to inhibit firing activity.

It is equally important to suppress the firing activity of undesired neurons to achieve a good classification performance. A novel depressive STDP rule is proposed which reduces the synaptic weight of undesired neuron when it fires a spike. This reduces the probability of firing of an undesired neuron in future.

3.4.1 CaL-S^{2}TDP Training algorithm

D-S^{2}TDP algorithm establishes the key idea behind supervised STDP learning in readout neurons. However, deterministic weight update causes several problems such as poor memory retention, weight saturation and large power consumption. To overcome these problems, Cal-S^{2}TDP algorithm is proposed.

Poor memory retention and weight saturation are attributed by the discrete levels of weights available when implementing in hardware. Frequent firing of desired neuron continuously increases the synaptic weight, resulting in weight saturation. No future weight updates are allowed, failing to capture information in the incoming spikes. This also results in high power consumption due to high rate of switching activity of several signals involved in various logic cells when a neuron fires frequently. To overcome this problem, a probabilistic weight update scheme is adopted which slows down the learning process providing a better learning performance. Saturation of weights during training is avoided by deactivating weight updates when a neuron either fires continuously or remains inactive. Activity of a neuron is determined by its internal calcium ion concentration. Calcium concentration of a neuron can be mathematically modeled as follows:

$$\frac{dc(t)}{dt} = -\frac{c(t)}{\tau_c} + \sum_i \delta(t - t_i) \tag{3.14}$$

where τ_c is the time constant and t_i is the time at which neuron spikes.

Based on the above considerations, CaL-S^{2}TDP algorithm is proposed as follows. Let C_T indicate calcium concentration threshold which separates an active neuron from an inactive one.

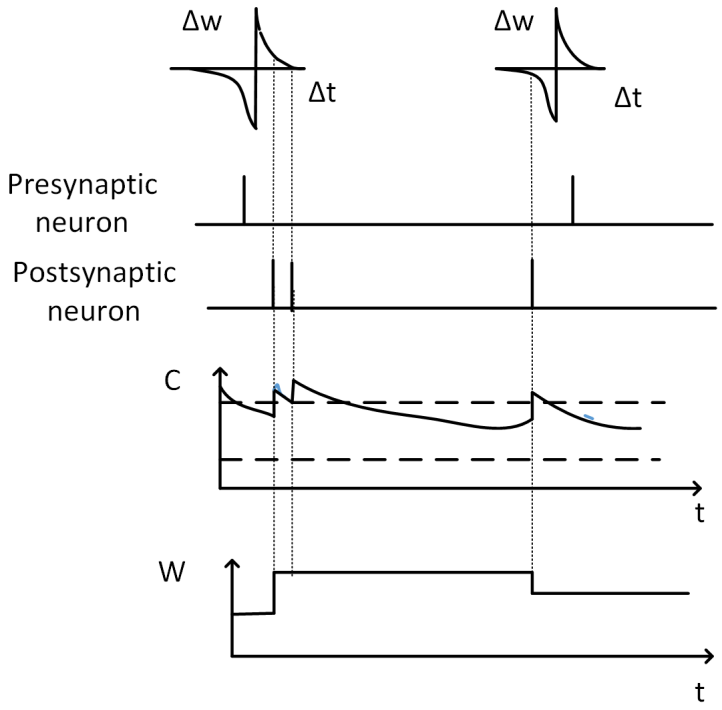

Figure 3.6: Training of desired neurons using CaL-S^{2}TDP

Let δ denote an activation margin. Synaptic potentiation is allowed if current calcium concentration of neuron lies in the range, $c_T < c < c_T + \delta$ and synaptic depression is allowed if internal calcium concentration lies in the range $c_T - \delta < c < c_T$. The weight updates are probabilistic with probability proportional to weight adjustments Δw. Following equation describes CaL-S^{2}TDP approach.

$$w = w + d, prob \propto \Delta w^+, \textit{ if } \Delta t > 0 \textit{ and } c_T < c < c_T + \delta \tag{3.15}$$

$$w = w - d, prob \propto \Delta w^-, \textit{ if } \Delta t < 0 \textit{ and } c_T - \delta < c < c_T \tag{3.16}$$

Figures 3.6 and 3.7 show the above learning process for desired and undesired neurons using CaL-S^{2}TDP. Positive correlation between presynaptic and postsynaptic spiking events strengthens synaptic weight and current induced through teacher signal invokes an additional spike. However, this does not allow increase in weight due to stochastic nature of the algorithm. A negative

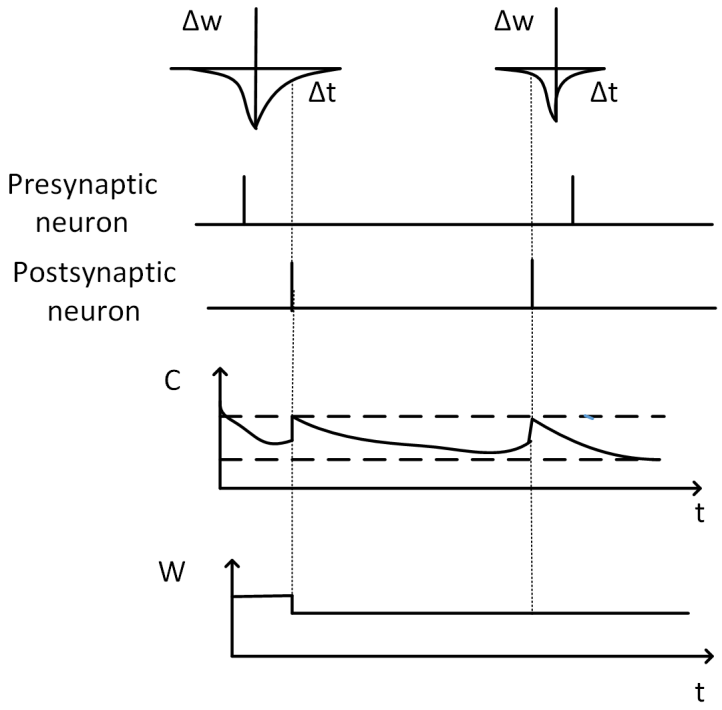

Figure 3.7: Training of undesired neurons using CaL-S^{2}TDP

correlation will reduce synaptic strength. In case of undesired neurons, a positive or negative correlation results in reduction of synaptic strength with teacher signal trying to inhibit firing activity. Decrease in synaptic strength is again stochastic in nature and thus, a decrease of weight is not observed in figure 3.7. Stochastic nature of the algorithm thus promotes a slow learning rate and prevents synaptic weight saturation.

3.4.2 Sparsification algorithm

The term sparsification, in this context, refers to reduction in the number of synapses between reservoir and readout neurons. Output layer in LSM is fully connected i.e each readout neuron is connected to every reservoir neuron. As the number of reservoir neurons are more compared to readout layer, reducing the synapses in a constructive way results in significant amount of energy savings. Having more number of synapses than required is also a sign of over-fitting. Thus, sparsification not only results in energy efficient architecture but also overcomes any possibility of

over-fitting the model. Sparsification can be achieved by allowing the readout synapses to compete among themselves based on the input firing patterns. This competition eliminates those synapses which are insignificant and because this elimination is based on input firing patterns, it will not have a significant affect on classification performance. Such competition can be induced by STDP learning rule. As each readout neuron is associated with a class label, it is only necessary to instruct each readout neuron to learn the sparse structure of input subset of its associated class. This leads to maximum sparsity and the information from other classes will not be mistakenly learned through the sparsification process.

CaS-S^{2}TDP algorithm for sparsification of readout synapses is proposed as follows. A sparsification teacher signal (ST) is introduced in the readout layer to bring up the firing activity of desired readout neuron so that initial random synaptic weight initialization will not affect sparsification process. ST signal also allows only synapses of desired readout neuron to participate in the sparsification process. A stop learning mechanism is also included similar to CaL-S^{2}TDP algorithm to avoid poor memory retention. CaS-S^{2}TDP algorithm can be summarized as follows:

$$w = w + d, prob \propto \Delta w^{+},\ \textit{if}\ \Delta t > 0\ \textit{and}\ c < c_T + \delta \tag{3.17}$$

$$w = w - d, prob \propto \Delta w^{-},\ \textit{if}\ \Delta t < 0\ \textit{and}\ c_T - \delta < c \tag{3.18}$$

The bounds on calcium concentration are relaxed in order to maximize sparsity as well as to avoid unnecessary bias in calcium regulation.

3.4.3 Two step training using sparsification and supervised STDP

Sparsification using CaS-S^{2}TDP and learning using CaL-S^{2}TDP are carried out in two steps. The network is first trained using CaS-S^{2}TDP algorithm which results in zero weight synapses in the readout layer. These synapses are removed and the remaining synapses are trained using CaS-S^{2}TDP algorithm starting with synaptic weights after sparsification training. As sparsification of synapses is carried out using input patterns, neural network captures spatio-temporal patterns in the input data set and hence using synaptic weights which are a result of sparsification, as

initial weights for classification results in a good performance. The weight update probability for supervised STDP learning algorithm is implemented using a look up table approach, similar to reservoir training. To minimize hardware overhead, same logic elements are used for sparsification and supervised STDP learning as they are carried out in non-overlapping time intervals. Detailed architectural details are discussed in Section 4.

4. HARDWARE ARCHITECTURE

This section describes the overall hardware architecture of an SNN processor along with implementation details of reservoir and readout layers.

4.1 Hardware architecture of SNN processor

Spiking neural network processor IP is developed on a Field Programmable Gate Array (FPGA). SNN IP is realized in the programmable fabric of FPGA which is interfaced with embedded ARM Cortex processors. It constitutes the processing sub system. IP communicates with processing sub system through ARM AMBA AXI ports. Data required for training and inference are provided to the processor IP through ARM interface. The control signals are used to control flow of data between processing subsystem and programmable fabric. Figure 4.1 shows the system architecture of SNN processor developed. Communication between ARM core and SNN IP is achieved through a handshake communication protocol. ARM processor transfers input spikes to SNN IP when the IP asserts a req_input signal. After a successful transfer of spikes, ARM core asserts an $input\_vld$ signal which indicates the IP that a new set of spikes are available for processing. IP then deasserts req_input signal after which the ARM processor deasserts $input\_vld$ signal. When SNN has processed input spikes and is ready to transfer these spikes to processing system, the IP asserts a $spike\_ready$ signal. When this signal is asserted, ARM core begins to receive data from SNN. When all the data has been received, ARM processor asserts $read\_ack$ signal. SNN IP then deasserts $spike\_ready$ signal and ARM processor then deasserts $read\_ack$ signal. After receiving output spikes from neural network,ARM processor determines which output neuron generated maximum number of spikes for a given input sample. The label associated with this neuron will be the inference made by neural network and this is compared with the ground truth.

Input spikes received from ARM processor are fed into reservoir layer. Each input spike is fed to a fixed number of reservoir neurons chosen in a random fashion. SNN IP consists of an input layer, reservoir layer and output layer. The section below describes reservoir architecture.

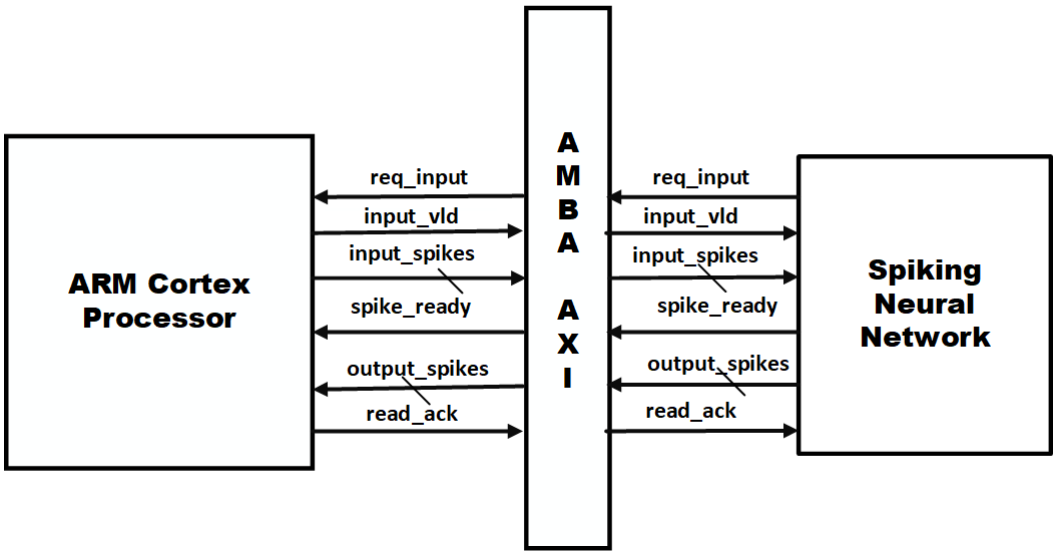

Figure 4.1: SNN Architecture

4.2 Reservoir Architecture

Reservoir is composed of randomly connected neurons resembling a liquid propagating disturbance originating at the input. Thus, these neurons are also called as liquid elements (LE). Each liquid element is composed of three modules, namely:

1. Reservoir neuron module
2. Reservoir synapse module
3. Reservoir learning module

Second order neuron and synapse models as described in Section 3 are implemented in neuron and synapse modules. Architecture of learning module varies in accordance with the learning mechanism used. Processing of an input spikes by a LE occurs in several stages which spans across several clock cycles. The number of clock cycles needed is a function of number of input as well as feedback connections to a reservoir neuron. The function of neuron module is to update membrane potential of neuron based on the state variables received from synapse and generate a spike if the integrated membrane potential exceeds threshold voltage (V_T). The spike thus generated is sent to readout layer, fed back to other reservoir neurons and is also buffered in a shift register

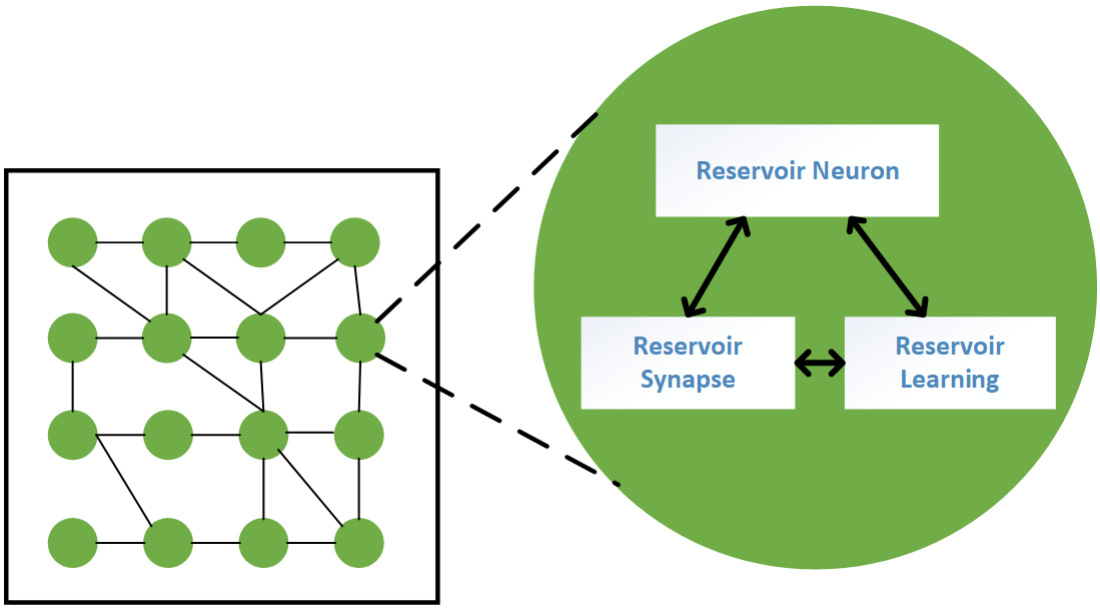

Figure 4.2: Reservoir Layer

for synapse weight updates. Figure 4.3 shows reservoir neuron architecture. The state variables EN, EP, IP, IN are received from synapse module. They are shifted right by an amount of time constant and then fed to an arithmetic unit along with current membrane potential. Arithmetic unit computes new membrane potential using updated state variables according to equation 2.8. This updated membrane potential is then compared with threshold voltage. If the new membrane potential is greater than threshold voltage, neuron generates logic 1 as its output. This bit is sent to the readout layer and is also buffered in shift register SR_0.

A reservoir synapse module takes as input, spikes from input layer along with reservoir feedback spikes and synaptic weight. This module updates state variables of neuron based on the equations described in section 2. This update of state variables also depends upon whether a spike received is from an excitatory or an inhibitory neuron. Figure 4.4 represents synapse module architecture. In this figure, X can be any one of the four state variables. In hardware implementation, there will be four such architectures, one for each of the state variables. The control signal $spike$ represents a spike coming from either an input neuron or one of the reservoir neurons.

Learning module implements an unsupervised STDP learning rule as described in Section 3.

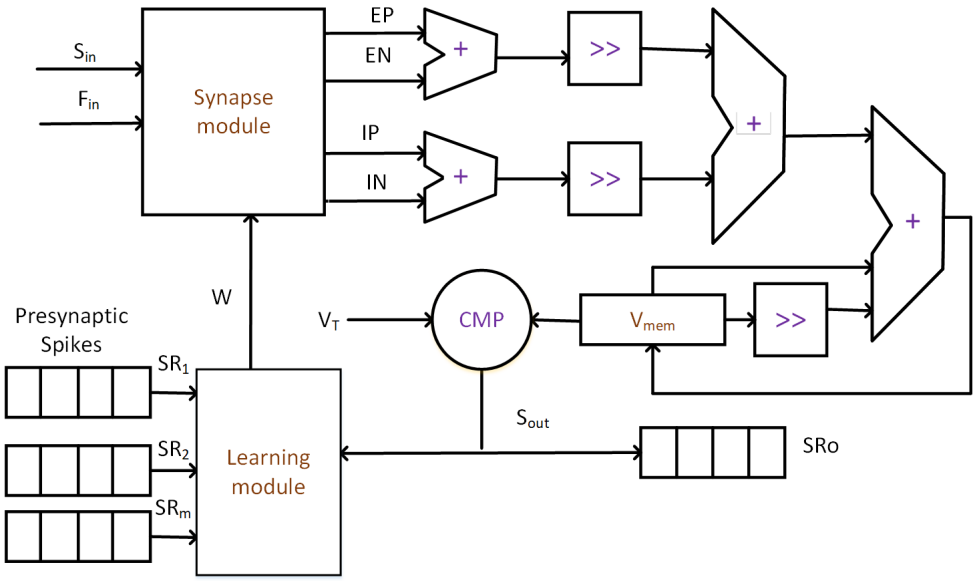

Figure 4.3: Reservoir neuron architecture

Synaptic weight update is based on a look up table which is indexed by the time difference between firing of pre-synaptic and post-synaptic neurons.Learning module provides updated synaptic weights to the synapse during learning phase. During inference, it is only neuron and synapse which remain active. Detailed description of learning module implementation is presented below.

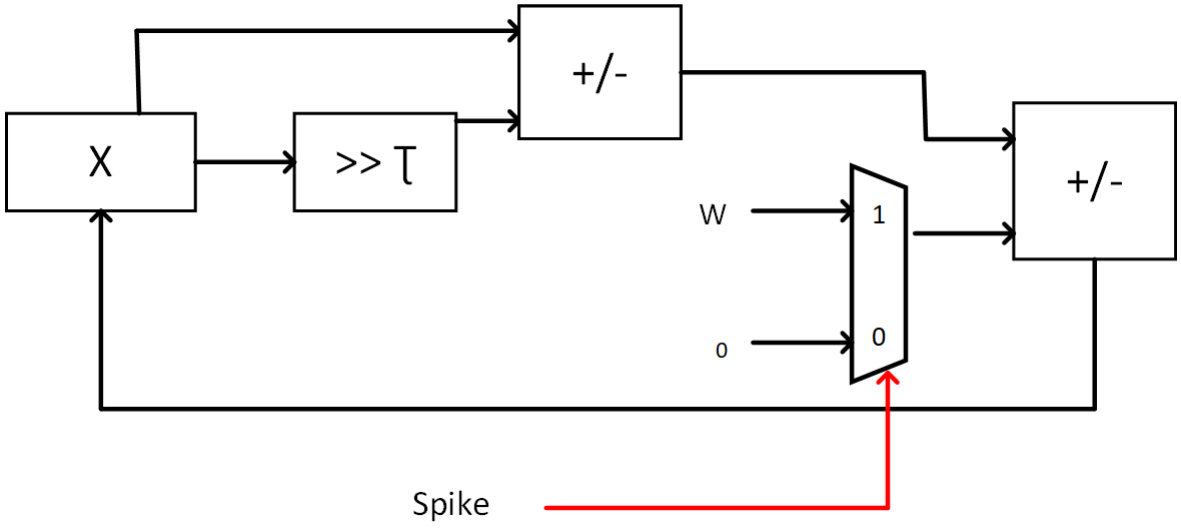

Figure 4.4: Synapse architecture

4.3 Readout Layer

The neurons in readout layer are termed as output elements (OE) as they are responsible for classifying input data and providing an inference. A major architectural difference between LE and OE is in the learning module, bit resolutions of synapses, state variables and memory to store synapse weights. Each OE uses a block memory (BRAM) to store weights of synapses associated with each of the reservoir neurons. LEs on the other hand make use of flip flops because of less number of synapses and lower bit resolutions. Similar to reservoir neuron, a readout neuron also consists of three modules:

1. Readout neuron module
2. Readout synapse module
3. Readout learning module

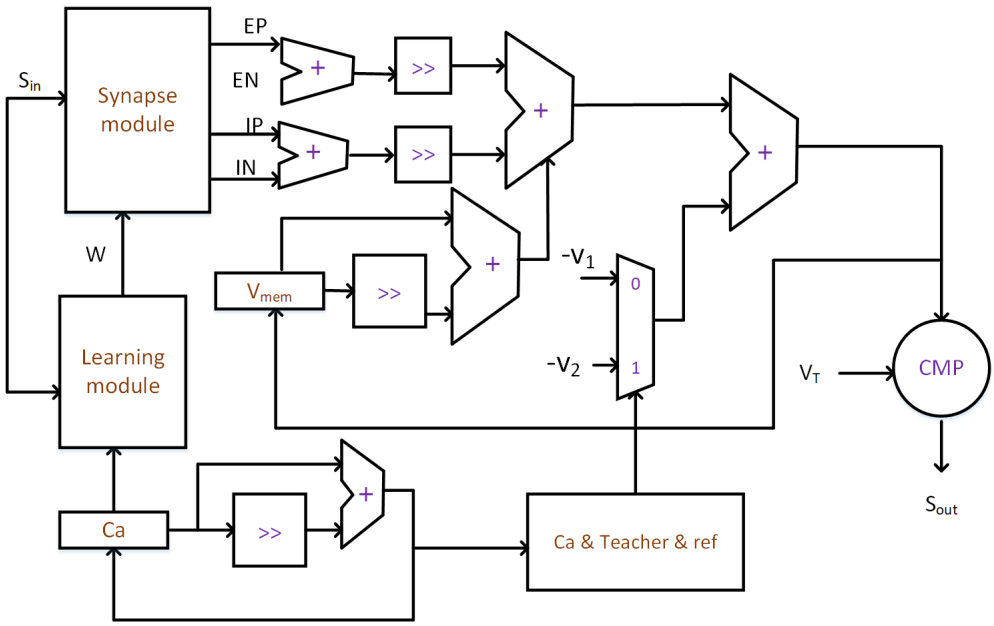

Figure 4.5: Readout neuron architecture

Figure 4.7 shows readout neuron architecture. Membrane potential of neuron is updated based on synapse state variables received from synapse module and current membrane potential of neu-

ron. Based on the calcium concentration of neuron and the refractory period, teacher signal adds additional voltage to the membrane potential. If the resulting voltage is greater than certain threshold, neuron generates a spike. Readout synapse module is similar to synapse module in reservoir and its architecture is same as shown in figure 4.6. Here spike control signal corresponds to a spike coming from a reservoir neuron. Detailed architectures of readout neuron and learning module are described below in section 4.5.

4.4 Implementation of unsupervised STDP

Learning module in LEs is responsible for tuning of plastic synapses between reservoir neurons. This module computes the time difference Δt_{ji} of spiking events between pre-synaptic and post-synaptic neurons utilizing shift registers. A post-synaptic neuron with m synapses tracks pre-synaptic events from m pre-synaptic shift registers $SR_1, ...SR_m$. The post-synaptic neuron itself has a shift register SR_0 in order to track the post-synaptic events. The depth of shift registers depend on the choice of time windows considered for LTP and LDP.

The neuron module updates the output shift register SR_0 when the post-synaptic neuron fires a spike. This spike is positioned at the MSB of the shift register as bit 1 and the remaining contents are shifted to the right by one position. In the absence of a spike, bit 0 is pushed into the MSB of the shift register. The contents of SR_0 are shifted right at each time step. The learning module calculates Δt by comparing the positions of the first spike from MSB in presynaptic and post-synaptic shift registers. Synaptic weight is updated only when there is a bit 1 in the MSB position of either presynaptic or post-synaptic shift register because it is an indication of a spike being fired by one of the neuron at that particular time step. For example, if shift register SR_0 has a spike in its MSB position and a presynaptic shift register SR_i has a spike in second position from MSB, then $\Delta t = 2$. On the other hand, if presyanptic register has a spike in MSB while there is a spike in second position from MSB in SR_0, then $\Delta t = -2$. Value of Δt calculated in this way is used to index LUT along with synaptic weight value. The value stoed in LUT location indexed by $(\Delta t, w_{prev})$ is used to compute new weight value w_{next}. This new synaptic weight value is then stored in weight memory and is provided to synapse module to update state variables.

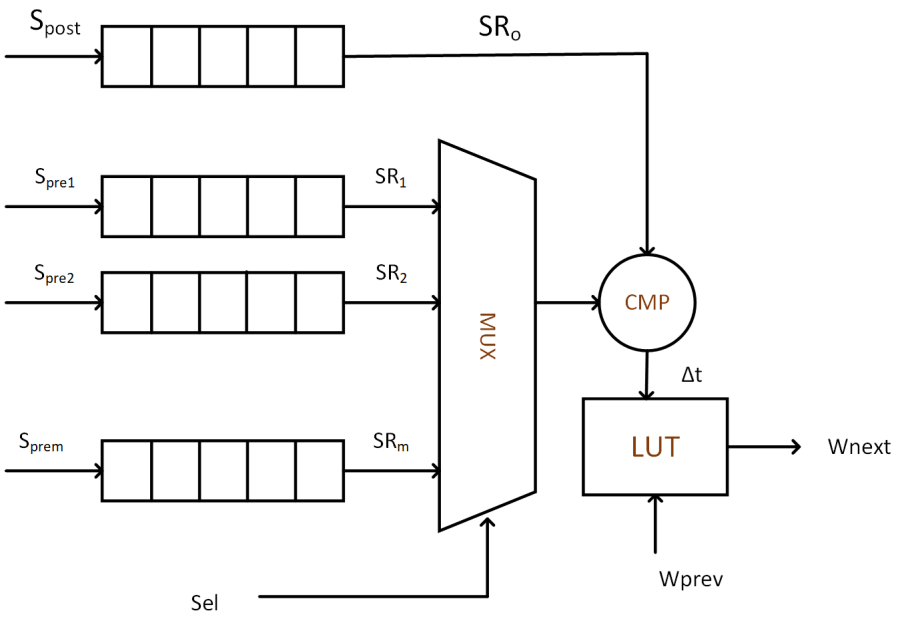

Figure 4.6: Unsupervised STDP Learning module

4.5 Implementation of supervised STDP

CaL-S^{2}TDP and CaS-S^{2}TDP learning and sparsification algorithms are fundamentally similar, although their function in the training process is different. Both these algorithms are based on STDP learning, involve probabilistic weight updates and calcium modulated stop-learning mechanism. The readout layer is trained with these algorithms in two separate phases and there is no overlap between their execution times. This gives an opportunity for sharing of resources across these two learning mechanisms leading to lower logic resource utilization and power consumption.

Figure 4.6 describes architecture of a readout neuron. Learning module of OEs has a similar architecture to the learning module of LEs that compute spike timing differences using shift registers. SR_0 is the output shift register of readout neuron and SR_1 to SR_m are pre-synaptic shift registers belonging to reservoir neurons. Due to full connectivity in the readout layer, the number of pre-synaptic shift registers connected to a readout neuron are high. sel signal selects one of the presynaptic shift register in each clock cycle and continues learning process until spikes from all presynaptic neurons have been processed, updating the weights associated with each of

the synapse. The sign bit of computed Δt determines if a potentiation or depression resulted. LTP and LDP look up tables store weight update probabilities. These probabilities are related to Δt. A smaller value of Δt indicates a high correlation between pre-synaptic and post-synaptic neurons and hence has a higher weight update probability. The entries in look up tables are hyperparameters which are tuned offline to improve performance. The LUTs are implemented with distributed RAM on FPGAs with zero read latency. The output from look up table is compared with the output of a pseudo random number generator. If the generated random number is smaller than the probability threshold in LUT, then weight is updated if the calcium concentration of readout neuron is in the range that allows learning. During readout sparsification, only synapses of readout neuron associated with class label participate in STDP tuning. In figure 4.6, signal CaS/CaL determines if the training phase is sparsification or supervised learning. During sparsification, path containing signal ST will be active and weights are updated only if signal ST is 1. In other words, sparsification takes place to the synapses associated with desired neurons. During learning phase, path containing signal CT is active and the weight update value is determined by the value of signal CT. In other words, classification teacher signal determines if a synapse weight has to be increased or decreased.

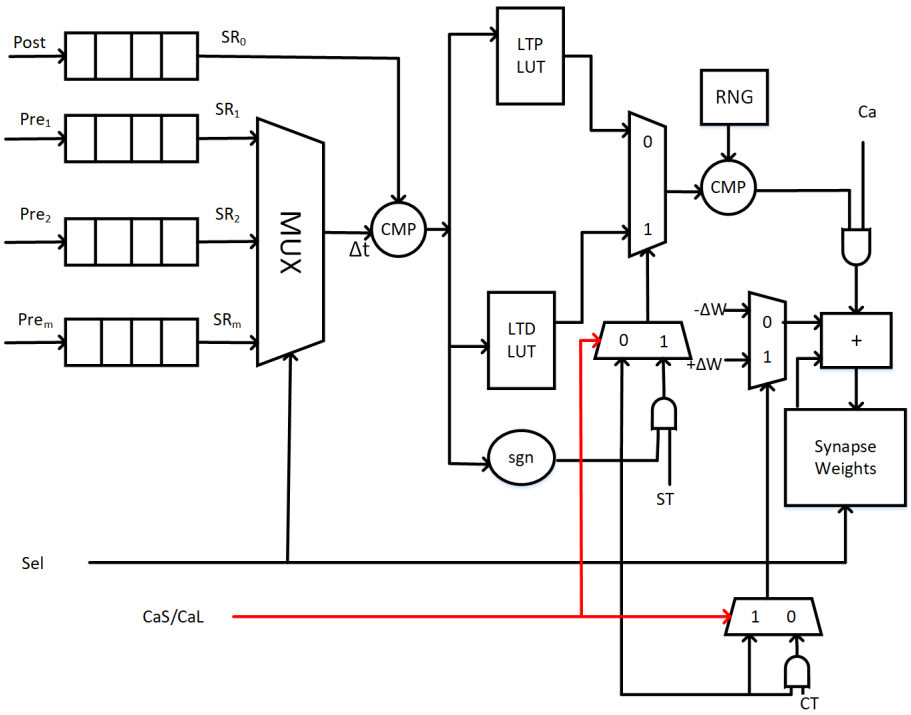

Figure 4.7: Learning in supervised STDP with sparsification

5.1 Experimental Settings and dataset

SNN processor described in Section 4 is designed and synthesized using Xilinx Vivado tool and targeted towards Zynq ZC706 FPGA board. Zynq ZC706 has an on chip ARM cortex processor which is used to provide data set to SNN processor along with control signals which are used to control different learning and testing phases. Data used to train and test performance of the developed processor is stored in an SD card which is interfaced to FPGA board. ARM processor receives data from SD card and provides to the neural network.

LSM architecture considered in this work has 78 neurons in the input layer, 135 neurons in the reservoir layer and 26 neurons in the output layer with each neuron being a representative of an input class. 80% of reservoir neurons are excitatory while remaining 20% are inhibitory. Reservoir synaptic weights are set to 2 and time window considered for weight updates in reservoir is 3. Table 5.1 shows look up table used for reservoir weight updates.

	$w_{prev} = 0$	$w_{prev} = 2$	$w_{prev} = 6$	$w_{prev} = 8$
$\Delta t = -3$	0	2	6	8
$\Delta t = -2$	0	0	2	6
$\Delta t = -1$	0	0	0	2
$\Delta t = 0$	0	2	6	8
$\Delta t = 1$	6	8	8	8
$\Delta t = 2$	2	6	8	8
$\Delta t = 3$	0	2	6	8

Table 5.1: LUT for unsupervised STDP in reservoir layer

Readout layer has synaptic weight resolution of 10 bits for all the learning rules used in this research to achieve optimal performance at low hardware overhead. These synaptic weights are initialized to random values in the range $[-2^9, 2^9 - 1]$. The depth of both LTP and LTD LUTs is set

to 16 and the entries of these LUTs are hyperparameters which are tuned offline to achieve good classification performance.

The architecture is trained and tested using a subset of TI Speech Corpus data set. This data set is a collection of utterances of 26 alphabets in English language. Each alphabet has a set of 10 different utterances with a total of 260 samples recorded from a single speaker. The speech signals obtained in time domain are first processed using Lyon's passive ear model [13] and then encoded into 78 spike trains using BSA algorithm [14]. Presence of a spike at a particular time instant is represented using logic 1 while absence of a spike is represented using logic 0. This data set is stored as a set of ASCII files on an SD card which is interfaced to FPGA board. A set of 8 ASCII characters represent the input spikes to 78 neurons in one time step.

Reservoir layer utilizing unsupervised STDP algorithm is trained using 20 iterations. It is observed that the learning process saturates after 20 iterations and they are sufficient for obtaining an optimal performance. Synaptic weights of reservoir layer obtained after 20 iterations are retained during training of readout layer. Readout synapses are trained using 250 iterations under different learning algorithms. Sparsification phase is carried out before readout training for 20 iterations. Zero weight synapses obtained after 20 iterations are disabled before moving on to readout training.

5.2 Classification performance

Classification performance is measured by determining the class of readout neuron which generated maximum number of spikes for given input pattern during testing phase. The spikes received from each of the readout neurons at every time step are received by the ARM processor which keeps a count of the spikes received so far from each neuron. Once a particular input class is completely processed, class inferenced by neural network is determined by the class of neuron with maximum activity. If this inferred class matches with the ground truth, it is counted as a success else a failure.

Five fold cross validation technique is used for training and testing the architecture developed. In this technique, a given dataset is divided into five groups and the whole process of training and testing is carried out five times. In each round, one group of data among 5 is used to test while

remaining four groups are used for training. A different group of data is used for testing the learned architecture in each round. Reported performance is an average achieved over all five rounds.Table 5.2 gives the classification performance of SNN processor employing different set of learning rules on TI Speech Corpus data set with 10 bit synapses in the readout layer. Figure 5.1 shows performance improvement achieved by different learning algorithms over baseline algorithm. In

Learning rule	**Performance**
Fixed + Base Line	91.53 %
Unsupervised STDP + Base line	93.46 %
Fixed + CaL-S^2TDP	94.23 %
Unsupervised STDP + CaL-S^2TDP	95 %
Fixed+ CaL-S^2TDP + CaS-S^2TDP	91.92 %
Unsupervised STDP + CaL-S^2TDP + CaS-S^2TDP	93.84 %

Table 5.2: Classification performance with 10-bit readout synapses

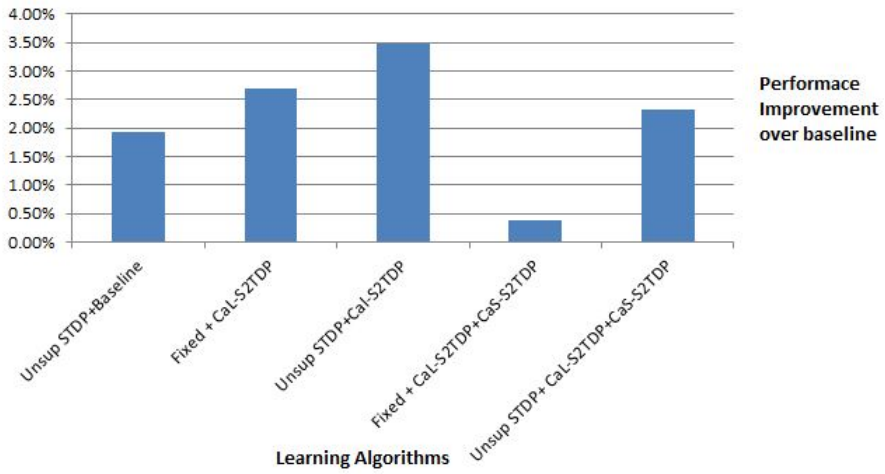

Figure 5.1: Performance improvement of various learning algorithms over baseline

table 5.2, the term fixed indicates that the reservoir synapses are fixed to +1 for excitatory synapses and -1 for inhibitory synapses. There are not trained using unsupervised STDP learning. Base line

indicates non-STDP supervised algorithm. Unsupervised STDP is the proposed hardware friendly algorithm as described in section 3. Sparsification achieved using $CaS - S^2TDP$ is about 25%.

From table 5.1, it can be observed that having supervised STDP in the readout layer and unsupervised STDP in the reservoir improves classification accuracy compared to having only fixed or unsupervised STDP in the reservoir. This performance benefits can be attributed to the tuning of synaptic weights based on correlations captured between presynaptic and postsynaptic neurons in both reservoir and readout layers. Hardware friendly unsupervised STDP over baseline provides a performance boost of 1.93 %. This indicates that STDP algorithm and LSM architecture are suitable for hardware implementation and provide good performance benefits in an optimally discritized environment. Having only supervised STDP in the readout layer provides a performance boost of 2.70 % over the base line. Turning the reservoir synapses plastic and training them using unsupervised STDP improves classification performance further providing 3.47% improvement over the base line. Upon the introduction of sparsification in the readout to improve energy efficiency, classification accuracy comes down by 2.31 % for the case of fixed reservoir and 1.16 % for the case of tunable reservoir. This decrease in performance can be attributed to reduction in number of synapses in readout layer making the network to over see some features in the input pattern. Given that performance degradation is not drastic, it can be concluded that these features are not crucial and can be ignored for lower power consumption. However, sparsification with fixed and tunable reservoirs provide higher performance compared to baseline processor. Having higher performance compared to baseline in the presence of sparsification indicates that removing synapes based on input patterns is more efficient compared to random removal of synapses.

Table 5.3 indicates classification performance of various learning algorithms with 8-bit synapses in the readout layer. From the results obtained, it can be observed that having a low bit resolution in the readout layer will have a degrading affect on the classification performance. It can also be observed from the table supervised STDP with fixed reservoir provides slightly higher performance compared to tunable reservoir. This can be attributed to the early saturation of weights in the readout layer due to lower bit resolution with reservoir layer trying to capture more features in

the input patterns through tuning of synapses using unsupervised STDP rule.

Learning rule	**Performance**
Fixed + Base Line	88.48 %
Unsupervised STDP + Base line	89.32 %
Fixed + CaL-S^{2}TDP	92.68 %
Unsupervised STDP + CaL-S^{2}TDP	91.534 %
Fixed+ CaL-S^{2}TDP + CaS-S^{2}TDP	90.38 %
Unsupervised STDP + CaL-S^{2}TDP + CaS-S^{2}TDP	90.76 %

Table 5.3: Classification performance with 8-bit readout synapses

Classification performance of processor is also determined by increasing synaptic weight resolution in readout layer to 16. However, it has been observed that the results are similar to processor with 10 bit weight resolution indicating that 10 bit is an optimal choice in terms of both classification performance and power consumption.

5.3 Hardware overhead

Tables 5.4-5.7 indicates hardware overhead involved in implementing each of the learning rules. The reported hardware utilization is of LSM processor alone. It does not include AXI

FFs utilization	12694 (2.90%)
LUT Utilization	43975 (20.18%)
BRAM	13(2.38%)
IO	90(24.86%)

Table 5.4: Hardware overhead for fixed + baseline

interface as the hardware overhead contributed by this interface is negligible. From the tables it can be observed that high performance of supervised STDP comes at a higher hardware overhead compared to rest of the learning algorithms. This hardware overhead is attributed to the use of

FFs utilization	12717 (2.91%)
LUT Utilization	45785 (20.95%)
BRAM	13(2.38%)
IO	90(24.86%)

Table 5.5: Hardware overhead for unsupervised STDP + baseline

FFs utilization	19841 (4.54%)
LUT Utilization	57581 (26.34%)
BRAM	13(2.38%)
IO	90(24.86%)

Table 5.6: Hardware overhead for unsupervised STDP + CaL-S^{2}TDP

FFs utilization	19844 (4.54%)
LUT Utilization	57788 (26.43%)
BRAM	13(2.38%)
IO	90(24.86%)

Table 5.7: Hardware overhead for unsupervised STDP + CaL-S^{2}TDP + CaS-S^{2}TDP

additional registers in the readout layer to compute spike timing difference for supervised STDP. The time window in the reservoir layer is set to 3 while that of read out has a time window of 12 to achieve reasonable performance gains. The depth of a shift register is equal to the time window and as there is a full connectivity between reservoir and readout neurons, the number of flip flops utilized is also higher in case of supervised STDP. These hardware overheads are not very high and are affordable for the performance gains achieved. Introduction of sparsification during learning reduces power consumption and also the increase in hardware overhead is very small. As the performance degradation is not very high, supervised STDP with sparsification provides a energy efficient solution to the development of spiking neural network architectures. Hardware utilization is only reported for 10 bit synaptic weight architecture in readout neurons as this architecture delivers good performance compared to and eight bit architecture.

5.4 Power Consumption

Tables 5.8 and 5.9 show dynamic power consumption for different learning algorithms for 10-bit an 8-bit readout synapse resolutions during training phase. These values are estimated using Xilinx Power Analyzer given activity based simulation results. The clock frequency used for calculating power is 100MHz consistent with the frequency of operation. From the tables it can be observed that training the processor with both unsupervised STDP in reservoir and supervised STDP in readout layer consumes highest amount of power compared to other learning mechanisms. Introduction of sparsification into this learning scheme reduces power consumption during training to 229mW from 237mW. This increase in power consumption is due to use of additional shift registers to train readout synapses using supervised STDP. However, use of sparsification reduces power consumption due to an increase in the number of inactive synapses that needs to be trained. This value includes some additional power consumed during sparsification of readout synapses. As logic elements are shared during sparsification and learning phases, power gains are achieved with the proposed architectural scheme. Lower power consumption values for 8-bit readout synapses are attributed to less number of flip flops needed to implement 8-bit shift registers in comparison to 10-bit shift registers. This reduces the number of bit toggling events leading to lower power consumption compared to 10 bit resolution of readout synapses.

Architecture	Power(mW)
Fixed + Base Line	181
Unsupervised STDP + Base line	195
Fixed + CaL-S^{2}TDP	210
Unsupervised STDP + CaL-S^{2}TDP	237
Fixed+ CaL-S^{2}TDP + CaS-S^{2}TDP	195
Unsupervised STDP +CaL-S^{2}TDP + CaS-S^{2}TDP	229

Table 5.8: Dynamic Power Consumption with 10 bit readout synapses

Arhitecture	Power(mW)
Fixed + Base Line	140
Unsupervised STDP + Base line	185
Fixed + CaL-S^{2}TDP	193
Unsupervised STDP + CaL-S^{2}TDP	220
Fixed+CaL-S^{2}TDP + CaS-S^{2}TDP	151
Unsupervised STDP + CaL-S^{2}TDP + CaS-S^{2}TDP	212

Table 5.9: Dynamic Power Consumption with 8 bit readout synapses

5.5 Conclusions

In this research, a hardware architecture with on chip learning capability is developed for implementing a spiking neural network processor. An efficient hardware platform using Xilinx Zynq ZC-706 FPGA is built to deploy neural network processor. Hardware friendly learning mechanism based on spike timing dependent learning plasticity is introduced. Supervised STDP for training readout synapses along with sparsification of readout synapses to achieve an energy efficient processor are incorporated into the processor architecture. Developed hardware is trained and tested using TI Speech Corpus dataset and performance of 95% is achieved without sparsification. It has been shown that sparsifying readout synapses using input features gives a reasonable performance with energy benefits and minimal hardware overhead.

REFERENCES

[1] "Ask a biologist. ©arizona board of regents,"

[2] "Shape of action potential." https://en.wikipedia.org/wiki/Action_potential. Accessed: 2018-09-18.

[3] D. J. Amit and Fusi, "Learning in neural networks with material synapses," *Neural Computation*, pp. 957–982, vol 6.

[4] D. J. Amit and S. Fusi, "Constraints on learning in dynamic synapses," *Network: Computation in Neural Systems*, pp. 443–464, 1992.

[5] B. A. Rou, Subhrajit. and A. Basu, "Liquid state machine with dendritically enhanced readout for low power, neuromorphic vlsi implementations," *IEEE Transactions on Biomedical Circuits and Systems*, p. Vol 8, 2014.

[6] L. Y. Jin, Y. and P. Li, "Sso-lsm: A sparse and self-organizing architecture for liquid state machine based neural processors," *IEEE/ACM International Symposium on Nanoscale Architectures*, pp. 55–60, 2016.

[7] Y. Jin and P. Li, "Calcium modulated supervised spike-timing-dependent-plasticity for readout training and sparsification of the liquid state machine," *International Joint Conference on Neural Networks*, pp. 2007–2014, 2017.

[8] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology,*, p. Vol 117, 1952.

[9] L. Y. Wang, Q. and P. Li, "Liquid state machine based pattern recognition on fpga with firing-activity dependent power gating and approximate computing," *Internatioal Symposium of Circuits and Systems (ISCAS)*, pp. 361–364, 2016.

[10] T. N. Wolfgang Mass and H. Markram, "Real time computing without stable states: a new framework for neural computation based on perturbations," *Neural Computation*, pp. 2531–2560, 2002.

[11] N. Caporale and Y. Dan, "Spike timing dependent plasticity: a hebbian learning rule," *Annual Revision of Neuroscience*, pp. 25–46, 2008.

[12] Y. Jin and P. Li, "Ap-stdp: A novel self-organizing mechanism for efficient reservoir computing.," *Neural Networks (IJCNN), International Joint Conference*, pp. 1158–1165, 2016.

[13] R. F. Lyon, "A computational model of filtering, detection, and compression in the cochlea," *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP82.*, pp. 1282–1285, 1982.

[14] B. Schrauwen and J. Van Campenhout, "Bsa, a fast and accurate spike train encoding scheme," *Proceedings of the International Joint Conference on Neural Networks*, pp. 2825–2830, 2003.